

BoxSoft
Corporation

Super QBE - Documentation

Copyright © 1989-2014, BoxSoft Corporation, All Rights Reserved.

Table of Contents

Foreword	0
Part I Getting Started	3
1 Introduction.....	3
2 RTFM Warning!!!.....	5
3 Installation.....	6
4 What is Query By Example?.....	10
5 What Are Tags? / Tag Storage Options.....	13
6 Upgrading from Earlier Versions.....	17
Part II Template Usage	20
1 Adding QBE to your Applications.....	20
2 Support Files.....	21
3 Global Extension Template.....	22
4 Browse Control.....	23
5 Form Control.....	25
6 Optionally Filter Tagged Records in a BrowseBox.....	31
7 Filter Tagged Records in a BrowseBox.....	33
8 Apply QBE Filter to Process/Report.....	34
9 Get Tagged Data Records in a Process/Report.....	35
10 Filter Tagged Records in a Process/Report.....	36
Part III Appendices	37
1 Printing Tagged Records with ReportWriter.....	37
2 Multi-APP Development using LIBs/DLLs.....	38
3 Interface Modification and Translation.....	40
4 Example Programs.....	41
5 Project Defines.....	42
6 API Reference - QBE.....	44
7 API Reference - Tagging.....	45
8 Troubleshooting.....	52
9 Contacting Technical Support.....	53
10 License Agreement.....	54
Index	55

1 Getting Started

1.1 Introduction

The SuperQBE Templates enable you to perform ad hoc query operations using Clarion. This will save you from writing numerous reports for your users. Instead, they can perform a wide variety of searches using their regular forms, then print the results using one of their standard reports. It's also handy for searching for records that the user "knows" exist, but they can't recall enough information to search for it using the normal browses.

QbeGlobal - This global Extension template adds the global data and map segments to your application. It also generates the QBE and Tagging modules. You control various general options which have an impact on the entire QBE system.

QbeBrowse - This Control template calls a Form procedure for a query operation. After the form returns, it automatically displays only the search results in your browse. (If your form has tagged the results, rather than merely computing the filter expression, then you must use the BrowseOptTagFilter control template for your browse to show only tagged records).

QbeForm - This Control template is placed on your Form. It performs the majority of the searching operations and interface control. It can tag all matching records, display each record for you optionally to edit it, or just compute the filter expression and return. You can also use the Wizard as an alternative to, or in conjunction with, the form.

QBEProcessFilter (*NEW in Super QBE 6.0*) - This Extension template can be used on a Process or Report. It applies the filter expression from the last search using the same primary file..

BrowseOptTagFilter - This Control template places a button on your window which will show "Tagged ", "Untagged" or "All" records. Then the user can control which set of records they wish to see by pressing this button.

BrowseTagFilter - This Extension template enables you to display only tagged records in a browse. This differs from the previous feature in that the filter is always "on".

ProcessTagFilter - This Extension template adds a filter to a Report or Process so that only tagged records are used by the procedure. This is similar to the BrowseTagFilter template, except that it is specifically used for Report and Process procedures, rather than BrowseBox windows. For a faster solution, see the "ProcessGetTaggedData" template.

ProcessGetTaggedData - This Extension template provides a faster alternative to the "ProcessTagFilter" template. With it, you process your report directly against TagFile_ or TagFilePos_ and this template will look up the tagged record. (If you are using your file's primary key field as the reference value stored by the tagging system, then you can use normal file relationships to perform this function instead of using this template.)

Tagging API - This collection of procedures and functions permit you full access to the tagging subsystem.

Multi-APP Support - Your systems can be comprised of a single APP/EXE, or many APPs/DLLs/EXEs.

The tagging library is contained only in your base APP.

Tags may be stored in a Memory Set, Disk Set, or by using a BYTE field in the primary file.

ABC and Legacy Template Chains

This documentation pertains to both the ABC and Legacy (a.k.a. "Clarion") Super Template sets. In some situations we've implemented features in ABC that are not in Legacy, primarily because the old template chain was to be phased out. Due to customer pressures, however, Soft Velocity decided to reinstate support for the Legacy/Clarion chain.

Some of the Super Template features that are only in the ABC chain would be very difficult to implement in the legacy chain. However, we'll attempt to do this wherever it seems feasible to us. We apologize if this causes you any inconvenience. Please feel free to contact us if there's a particular feature in ABC that you would like to see in the Legacy chain, and we'll see if your needs can be accommodated.

For more information, see the following topics:

- What is Query By Example
- What Are Tags? / Tag Storage Options
- Upgrading From Earlier Versions
- Adding SuperQBE to your Application
 - Support Files
 - Global Extension Template
 - QBE Browse Control
 - QBE Form Control
 - Optionally Filter Tagged Record in a BrowseBox
 - Filter Tagged Records in a BrowseBox
 - Get Tagged Record in Process/Report
 - Filter Tagged Records in a Process or Report
- Printing Tagged Records with ReportWriter
- Multi-APP Development using LIBs/DLLs
- Interface Modification and Translation
- Example Programs
- API Reference
 - QBE
 - Tagging
- Troubleshooting
- Contacting Technical Support
- License Agreement

1.2 RTFM Warning!!!

It is very important that you read this documentation. If you follow the instructions step-by-step, then the usage is very simple. It is almost *impossible* if you try to do it on your own!

1.3 Installation

Installation Directory Structure

NOTE: As of version 6.6, we've changed our installation to the defacto "3rdParty" directory structure. (In Clarion 7 this is actually the "Accessory" directory.) Your old CLARIONx\SUPER directory has been renamed to CLARIONx\SUPER-OLD.

Once you've finished running the installation program, you should see the following structure under your C55 or Clarion6 directory:

```

C:\CLARION6, C:\C55, etc.
+-LibSrc      STA*.INC      (ABC headers)
+-3rdParty
| +-Bin ST_*.HLP, ST_*.CNT, STAB_CNV.DLL
| +-Template  STA?*.TPL, STA*.TPW      (ABC chain)
| |          STC?*.TPL, STC*.TPW      (Clarion chain)
| +-LibSrc    STA*.INC, STA*.CLW, STA*.TRN (ABC chain)
| |          STC*.INC, STC*.CLW, STC*.TRN (Clarion chain)
+-Images
| `--Super   *.ICO, *.CUR, *.WMF, *.GIF
+-Docs
| `--Super   *.PDF      (Documentation)
`--Vendor
   `--Super
      +-QBE
      | +-Examples
      | | +-ABC *.DCT, *.APP, *.TPS (Examples) (ABC chain)
      | | `--Clarion *.DCT, *.APP, *.TPS (Examples) (Clarion chain)
      | `--Source
      | | +-ABC *.TXD, *.TXA, *.DCT (Source) (ABC chain)
      | | `--Clarion *.TXD, *.TXA, *.DCT (Source) (Clarion chain)
      +-Tagging
      | +-Examples
      | | +-ABC *.DCT, *.APP, *.TPS (Examples) (ABC chain)
      | | `--Clarion *.DCT, *.APP, *.TPS (Examples) (Clarion chain)
      | `--Source
      | | +-ABC *.TXD, *.TXA, *.DCT (Source) (ABC chain)
      | | `--Clarion *.TXD, *.TXA, *.DCT (Source) (Clarion chain)
      `--Etc.
      +- . . .
`--SUPER-OLD      (ABC headers)
   `-- . . .

```

For Clarion 7 and later versions it should look like this (note the two trees):

```

C:\Program Files\SoftVelocity\Clarion 7
`--Accessory
   +-Bin          ST_*.HLP, ST_*.CNT, STAB_CNV.DLL
   +-Template
   | `--Win       STA?*.TPL, STA*.TPW      (ABC chain)
   |              STC?*.TPL, STC*.TPW      (Clarion chain)
   |              STMH*.TPW                (Shared)
   +-LibSrc
   | `--Win       STA*.INC, STA*.CLW, STA*.TRN (ABC chain)
   |              STC*.INC, STC*.CLW, STC*.TRN (Clarion chain)
   +-Images
   | `--Super     *.ICO, *.CUR, *.WMF, *.GIF
   `--Docs
      `--Super     *.PDF      (Documentation)

```

```

"My Documents" or "Shared Data" (depending on OS)
^-Clarion 7\Accessory
  ^-Super
    +-QBE
      | +-Examples
      | | +-ABC          *.DCT, *.APP, *.TPS (Examples)      (ABC chain)
      | | ^-Clarion     *.DCT, *.APP, *.TPS (Examples)      (Clarion chain)
      | | ^-Source
      | | +-ABC          *.TXD, *.TXA, *.DCT (Source)         (ABC chain)
      | | ^-Clarion     *.TXD, *.TXA, *.DCT (Source)         (Clarion chain)
    +-Tagging
      | +-Examples
      | | +-ABC          *.DCT, *.APP, *.TPS (Examples)      (ABC chain)
      | | ^-Clarion     *.DCT, *.APP, *.TPS (Examples)      (Clarion chain)
      | | ^-Source
      | | +-ABC          *.TXD, *.TXA, *.DCT (Source)         (ABC chain)
      | | ^-Clarion     *.TXD, *.TXA, *.DCT (Source)         (Clarion chain)
    ^-Etc.
    +- . . .

```

To prevent conflicts between old Super Template files and same-named files in our new directory structure, the new installers attempt to delete the old files. If it encounters problems, then an error will be reported during the installation. Then you must delete any of the following files from the old directories, if they also exist in the new directory structure:

```

C:\CLARION6, C:\C55, etc.
+-LibSrc          STAB*.CLW, STCL*.CLW, STAM*.CLW, STCM*.CLW,
|                STAB*.TRN, STCL*.TRN, STAM*.TRN, STCM*.TRN,
|                STDEBUG.*
+-Template       STAB*.TP?, STCL*.TP?, STAM*.TPW, STCM*.TPW,
|                STGROUPS.TPW, STDEBUG.TPW
^-Bin            ST_*.HLP, ST_*.CNT, ST_*.GID

```

For example, you can use a tool like the indispensable Beyond Compare (www.scootersoftware.com) to investigate the contents of C:\Clarion6\LibSrc and C:\Clarion6\3rdParty\LibSrc. View only files matching the mask ST*.* and hide all "orphans", which will show the files that exist in both directories. Delete the files from C:\Clarion6\LibSrc, and then do the same for the Template and Bin directories.

Filenames and Product Abbreviations

- Super Template filenames generally start with the letters "ST". That's about all you can go on most of the time. (Our image files don't follow this convention, but they are sequestered in the Image\Super subdirectory.)
- The next two letters are usually AB (for the ABC chain) or CL (for the Clarion/legacy chain). One exception is Super Stuff (MH), which uses AM, CM and MH. Also, if both the ABC and Clarion chain share a TPW, then the AB/CL are skipped and it goes on to the product abbreviation. (Again, Super Stuff is an exception, as it uses MH for the shared files.)
- There are several TPWs that are shared by multiple Super Templates: STGROUPS.TPW, STABABC.TPW, STBLDEXP.TPW, STDEBUG.TPW
- The last four characters:
 - For TPL files, the last four letters are an underscore, followed by one of the following

suffices. The exceptions are STABAEQB.TPL and ST?M_STF.TPL.

- For TPW files, the last four letters may match one of these in its entirety, or be followed by additional characters denoting the special purpose files.
- Super Stuff (MH) is an exception, in that it uses STcMxxxx, where "c" is the chain of A or C, and "xxxx" denotes the special purpose.

AEQB	Super QuickBooks-Export (i.e. Accounting-Export QuickBooks)
BRW/BW	Super Browse
DIA	Super Dialer
FF	Super Field-Filler
IE	Super Import-Export
INV	Super Invoice
LIM	Super Limiter
PCD	Super Passcode
QBE	Super QBE
SEC	Super Security
TAG	Super Tagging
MH/STF	Super Stuff (MH) (a.k.a. <i>The "MikeHanson" Templates</i>)

Update the Redirection File

The installation program is able to update your redirection file automatically. If you decline the option during the installation, then you will have to edit the redirection file yourself. The three things that must be found are the Templates, LibSrc and Images. For example, you might make the following changes to the the *.* entry in Clarion 6:

```
*.* = .; %ROOT%\examples; %ROOT%\libsrc; %ROOT%\images; %ROOT%\template; %ROOT%
      \3rdParty\template; %ROOT%\3rdParty\libsrc; %ROOT%\3rdParty\images\super
```

In Clarion 7 and above it will be more like this:

```
*.* = %ROOT%\Accessory\images; %ROOT%\Accessory\resources; %ROOT%\Accessory\libsrc\win; %
      ROOT%\Accessory\template\win; %ROOT%\Accessory\images\Super
```

There are *.RED examples in the SUPER\DOC directory.

Register the Templates

Clarion allows you to have multiple template sets accessible in the same application. It does this with the Template Registry. To use a Super Template, you must register it first. The installation program attempts to do this for you, but in case it fails, or if your registry becomes corrupted, then you must register them manually.

1. Load Clarion, then select the "Setup / Template Registry" pulldown menu option.
2. Press the [Register] button.
3. Select C:\CLARION\3rdParty\Template\ST_*.TPL (ABC) or ST_*.TPL (Clarion). The directory name may not exactly match your system.

Assuming this all went without a hitch, you're ready to start using the templates.

1.4 What is Query By Example?

Query By Example, or "QBE", is used to perform ad hoc searches for records in your data files. It uses a regular Form to input the search criteria, and/or a Wizard. The query results can be edited, or selected for subsequent tagging operations. You can use a variety of comparison operators on each field, including "exact match", "doesn't equal this", "contains this", "doesn't contain this", "begins with this", "ends with this", etc. You can also specify multiple values for a single field, and search for data in related files.

Normally a QBE Form is called from a QBE button on a Browse, although it can be called from your own hand-coded procedures. (Just set `GlobalRequest=QbeRequest` before calling the Form.) The QBE button on the browse usually has display text of "Search". This button is populated onto the Browse using the `QbeBrowse` Control template.

The Form must have the `QbeForm` Control template. This creates two buttons: "Search Type" and "Begin Search". Depending on the type of entry field, a different default comparison operator (Search Type) is assumed. The default is "begins with" for a string field, "equals this" for a number field, etc. You have control of the defaults for String, Number, Picture, Date, Option, Check, and Text fields. You can modify these defaults in the template settings.

If you have turned the Wizard interface on, the user will be presented with 5 steps:

1. Introduction
2. Load Criteria Set
3. Specify Criteria
4. Save Criteria Set
5. Finish

If you have the interface set for Wizard+Regular, the Form window will appear once the user presses the Finish button. At that time they may tweak their query further.

There is a corresponding "flag" button to the left of each search field that will show the comparison operator for the field. Make sure that the field isn't positioned closely after another field or PROMPT(). If it is, the existing controls will be overlaid by the comparison flags. The various display symbols are defined as icons in the `SUPER\ICONS` directory.

If the user wishes to override this default for a given field, they must press the "Search Type" button, right-click on the field, or click the flag button to the left of the field to see a list of possible comparison types. This list is context sensitive: it displays only the applicable comparison types for the data type of the field. They will see all the criteria for this field in the list box. They can insert, change and delete the criteria here. If they change the value in the original window, it affects only the first criterion.

While they are in the criteria screen, they can save the load criteria sets for the entire form. They can optionally overwrite or retain the existing criteria during each of these operations.

If the user enters more than one field in the form, the comparisons are joined in a Boolean "AND" operation. For example, if the user specifies a last name of "Smith", a city of "Toronto", and a salary of more than "\$50,000", they will get only those records matching all three of those criteria.

If they specify multiple criteria values in a single field, they will be handled with an "OR" operation like: records where the last name is "Smith" or "Jones", and they live in "Toronto" and ...

Once they have finished entering the criteria, they can push the "Begin Search" button, then choose to edit the results, or to select all matching records. This will actually vary depending on how you've set up the prompts. (For the time being, we'll assume that all options are enabled.)

If the user chooses to "Edit" the records, each matching record will be displayed in the form. The "Search Type" button is changed to "Edit This", while the "Begin Search" button is changed to "Find Next". The "Cancel" button can be used to abort the search. Once they've accepted the first record they are immediately placed into edit mode so that they can make any desired changes. After finishing the update, the form immediately returns to the calling procedure (i.e.: the Browse).

If you have the interface set to "Wizard" (not "Regular" or "Wizard+Regular"), then the Edit option will not be available.

If you have turned on the "Select Support", and if the user chooses to "Select" the records, then all matching entries will be tagged. You can use the BrowseOptTagFilter, BrowseTagFilter, ProcessTagFilter, and ProcessGetTaggedData templates to restrict which records are available. If you have our SuperTagging templates, you can also modify the selections using those tagging features. Make sure that all of the associated procedures use the same tag storage settings.

NEW in Super QBE 6.0 (ABC): You can have the Form build the filter expression, and then pass it back to the calling Browse, which can then use it for its filter. Of course, this negates the benefits of tagged result sets, but it's much faster if you merely want to see a list of matching records.

If there are already tagged records from a previous search, then the user will be given an additional option of "Untagging". This will limit the search only to the previously-tagged records. This is handy if you want to find a bunch of records with one operation, then selectively remove a few with a series of untag operations.

For example, you may want to determine all people from Toronto who don't live in an apartment. You would first select all people who live in Toronto. Then you would perform a second query to unselect those with "APT", "SUITE" or "STE" in their address.

If there are previous tags but the user chooses to Select the records, they will be given the option to clear the previous selections first. Depending on the type of data in your file, they may have handled the "people in Toronto not in an apartment" example differently. Rather than performing a Select followed by various Unselects, they could have done multiple Select operations. The first would get all people in Toronto where the address didn't contain "APT". Each subsequent search operation could add to the previous search's results.

The file access is performed using a View structure with a Filter. If the user specifies criteria for fields in a related file, then the corresponding record in the primary file will be selected. The View structure optimizes searches for keys in the primary file. For example, if your primary file is Customer with Invoice as a secondary, searches for "LastName begins with 'H' " will be fast, while searches for customers with invoices in a particular date range will be slower.

Valid comparison operators include "exact match", "begins with", "in range", "more than", "less than", "more than or equal", and "less than or equal". With this in mind, you may want to reorganize the keys in your data files to optimize your searches. For example, if you anticipate that the user will perform many searches involving salary ranges, then you might make the first key in your file based upon the salary field. If you specify multiple values for a keyed field, the View will not use the key.

1.5 What Are Tags? / Tag Storage Options

You will often be asked to specify your Tag Storage options. For a particular file, you should be as consistent as possible when entering these. Of course, there will be exceptions when you normally store tags in memory, but you want to save them to disk.

NOTE: For the Memory and Disk Set tag storage methods, the tags are stored separately from your data file, with a pointer back to your record's primary key field. This means that your data file must contain a key marked as Unique and Primary. The key must contain only one field component. Either an integer (e.g.: LONG) or a string will do. Of course, a LONG is preferable, in which case the key should also be Auto-Numbered.

If you *cannot* mark one of your keys as Primary, then a suitable Unique key can be used instead. To tell Super Tagging to use the key, add the following setting to the key's options in the dictionary:

What is a "Tag Set"

Think of it as a collection of tags associated with a particular file and used for a certain purpose. You should use a different Tag Set number for each data file; otherwise, there is a very good chance that your tag sets will corrupt each other. If you need two different types of tags associated with the same file, just use two different tag set numbers.

This can be a constant (e.g. 1), an equate (e.g. eT_Customer or Tag:Customer), or a variable (e.g. Loc:TodayTags). It's a good idea to define tag sets as global equates, rather than integer constants. For example:

```

ITEMIZE(1),PRE(Tag)
Customer      EQUATE
Employee      EQUATE
Product       EQUATE
Invoice       EQUATE
EmpInvoice    EQUATE ! separate invoice tag set
END!ITEMIZE

```

Tag Storage Settings in Clarion/Legacy versus ABC

In the ABC templates, you have the option of:

- Memory Set

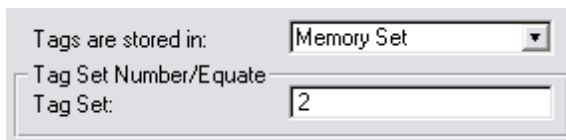
- Disk Set
- BYTE Field in Primary

The legacy templates differ, in that they offer:

- Memory (POINTER)
- Memory (POSITION)
- TagFile_ (POINTER) i.e. "Disk"
- TagFilePos_ (POSITION) i.e. "Disk"
- BYTE Field in Primary

The key difference is that the ABC templates automatically determine whether your primary key is an integer (e.g. LONG). If it's an integer, it uses the "POINTER" method, which are really just whole number tag references. Otherwise, it uses the "POSITION" method, which stores the tag reference as STRINGS.

Memory Set



The screenshot shows a dialog box with the following fields:

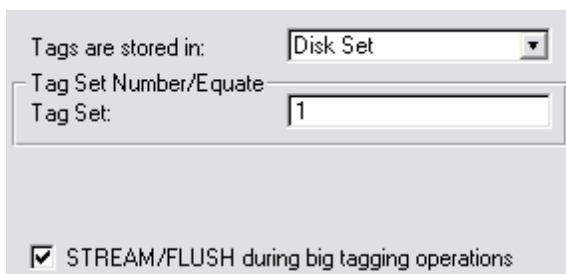
- Tags are stored in: Memory Set (dropdown menu)
- Tag Set Number/Equate: 2 (text input field)

Using this tag storage method, the tags are stored in a memory queue. Each entry contains a LONG integer that equals your data record's primary key field.

Because the tags are stored in memory, multiple users can tag records without interfering with each other. There are two disadvantages to this method:

1. The tags are only static while the program is running. If you need to save the tags from one session to the next, you can use the SaveTags or CopyTags templates, or you can use "Disk Set" instead.
2. If you are tagging only a few records of a large data file, and you wish to process only the tagged records, using Memory Set forces you to use a slow filter rather than a fast range. Again, you could copy the tags to a "Disk Set" before running the Process or Report, but this can get a little confusing.

Disk Set



The screenshot shows a dialog box with the following fields:

- Tags are stored in: Disk Set (dropdown menu)
- Tag Set Number/Equate: 1 (text input field)
- STREAM/FLUSH during big tagging operations

The difference between this and "Memory Set" is that the tags are stored on disk. It's slower, but

the tags are remembered from session to session. You can also use it directly for Reports and Processes, looking-up the related data record as each tag is processed.

NOTE: If you wish to process tagged records in ReportWriter, you will need to save them on disk.

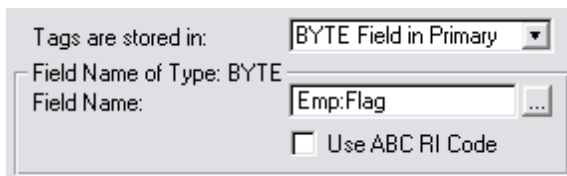
Tags are stored in either TagFile_ or TagFilePos_, depending on whether your primary key field is a numeric or string field. TagFile_ and TagFilePos_ are separate files usually using the TopSpeed file driver (or other driver of your choice).

Each record contains the user number, tag set number, primary key, and sorting value. This enables you to have multiple users tagging records in the file at the same time. Or a single user could create multiple tag sets for the same file (e.g.: one for billing and another for advertising).

The main disadvantage to this method is the slowness with which tags are added and deleted. However, if you need to report on a small number of tagged records from a large data file, this is the preferable method.

STREAM/FLUSH during big tagging operations (not always present) - If only one person is normally using the system, then you may want to turn this on. It locks the TagFile during these operations, so you shouldn't use the feature in a multi-user environment unless you don't have many users tagging at the same time.

BYTE Field in Primary



Tags are stored in: BYTE Field in Primary

Field Name of Type: BYTE

Field Name: Emp:Flag

Use ABC RI Code

This tag storage method toggles a BYTE field in your primary file. It is not multi-user compatible, but this may not be an issue for you. It happens to be the easiest to use in Reports and Processes (especially ReportWriter). It is not, however, the most efficient. In some ways it is faster than using Disk Set, and in some ways slower.

You may wish to use one of the other tagging methods for the actually tagging itself, then use the CopyTags code template to save the tags in the BYTE field before running a Report or Process.

Field Name - This is the field that will hold the "Tagged" status.

Tagged Value (*not always present*) - This is the value assigned to your "BYTE Field in Primary" when the record is tagged. It applies only if you are using that storage method. You can specify a number, string or variable name (without preceding exclamation). The value is used without modification (e.g.: Pre:TagField = YourSetting). The default is "True".

STREAM/FLUSH during big tagging operations (*not always present*) - If only one person is normally using the system, then you may want to turn this on. It locks the primary data file during these operations, so you shouldn't use the feature in a multi-user environment.

1.6 Upgrading from Earlier Versions

Upgrading from Super QBE pre-4.0

Changes in QBE

There are relatively few changes in SuperQBE with the 4.0 release. They are as follows:

- If you had not already imported the QBE support files into your dictionary, you must do so now. The import file is SUPER\LIBSRC\QBE\QBE.TXD. If you have already imported them, then the file labels and prefixes have been changed, and you must make a similar change in your existing dictionary. (The physical file names on disk remain the same: QBESET_ and QBECRIT_.)

OLD	NEW
SQBE::CriteriaSetFile,PRE(QS_)	SQBE::Set,PRE(QSet_)
SQBE::CriterionFile,PRE(QC_)	SQBE::Criterion,PRE(QCrit_)

- There are a number of new settings in the Global Extension.
- The Frame extension is now extinct. It should be removed from any windows where it exists. (If you forget to remove them all, the template will remind you when the code is generated.)
- A number of equates have been changed. (This will only affect you if you are hand-coding calls to QBE forms.)

OLD	NEW
QbeAction	QbeRequest
eQbeEditResponse	QbeResponse:Edit
eQbeSelectedResponse	QbeResponse:Selected

Changes in Tagging

We have made a number of changes to the SuperTagging templates during our migration to Clarion 4+ABC. Please take note of the following:

- If you had not already imported the tagging files into your dictionary, you must do so now. The import file is SUPER\LIBSRC\TAGGING\TAGGING.TXD.
- We are no longer using Glo::Pointer and Glo::Position. You can delete these variables from your Global Data.

Tag Storage Changes

We first added tagging features to the SuperModels for Clarion 2.0 for DOS. At that time only Clarion datafiles were supported and the concept of a "primary key" wasn't very common in the Clarion world. Therefore, we decided to use POINTER(File) instead. With the addition database

drivers in Clarion 3.0 for DOS, we added support for POSITION(File). In our templates for Clarion for Windows, we added BYTE Field in Primary. After this, we also added support for using both numeric and string primary key fields, and also for POSITION(PrimaryKey). Finally, we added support for storing these tags in Memory.

With the release of SuperQBE 4.0 and SuperTagging 4.0, we have reduced these storage methods from eleven to five, and we simplified their usage!

Clarion 4 (ABC) has enforced the use of VIEWS as the preferred method of file access. For us to support POINTER(File), POSITION(File) and POSITION(PrimaryKey), we would have had to make excessive use of the REGEX operation. At the very least, this would have made tagging operations very slow. Instead, we decided that support for these methods is not longer necessary.

Because of this decision, you will be required to have a primary key in your file with a single field component. (The inclusion of a primary key of this type is good database practice, anyway.) This primary key field can be an integer (e.g.: LONG) or a string. If the field is an integer, then tags will be stored in TagFile_ . Otherwise, they will be stored in TagFilePos_ .

When you run your APP through Clarion's Conversion Wizard, it will automatically make the modifications to your tag storage settings. "TagFile_ (POINTER)" and "TagFilePos_ (POSITION)" are changed to "Disk Set"; "Memory (POINTER)" and "Memory (POSITION)" are changed to "Memory Set"; while "BYTE Field in Primary" is left as-is.

NOTE: If you don't care about existing saved tags, you can just delete TAGFILE_.TPS, TAGFILEP.TPS and TAGSET_.TPS. If you have always set "Reference Source" to "Primary Key", then your tags are already compatible with the new system. If neither of these applies, then you will have to create a tag conversion program.

The conversion program is created by a Utility template included with SuperQBE 2.5x and SuperTagging 2.5x. Unfortunately, the utility templates cannot access all of your various settings, so you will have to help it along.

Start by loading your application with the legacy templates. Press Ctrl-U (or select "Application / Template Utility" from the pulldown). You'll find two template utilities under "Class SuperQBE":

TagTemplateFinder - This Utility template is used to list all procedures which incorporate tagging operations, so that you can record those settings and enter them into the TagStorageConvertor.

TagStorageConvertor - This Utility template creates a PRJ+CLW to convert your tags from unsupported legacy storage methods to the new methods. It automatically scans your APP for a list of the datafiles involved in tagging operations. You must specify the tag storage method for each of these files, plus any tag set numbers that need to be converted. Saved tags are handled for all specified files.

NOTE: This conversion program assumes that you have been consistent with your tag storage settings:

- You must never use the same tag set number for two different files.

- For a particular file, you must specify a consistent setting for "Tags are stored in", "Reference Source", and "Use POSITION(PrimaryKey)". Otherwise, the convertor will jumble your tags.

The template will use your global extension setting for "Tag File Location" to determine whether it should generate the tag support files itself, or produce them from the dictionary.

Once you've created the conversion program, you need only load the project and compile it into an EXE. Of course, you should tag some records in the legacy version of your program, run it through the conversion wizard, then check it against the converted tags.

2 Template Usage

2.1 Adding QBE to your Applications

The SuperQBE templates are actually a combination of templates and functions. The templates should handle all interfacing with the functions, so you should never have to call them yourself. If you are adventurous, though, you can certainly give it a try. (For more information on this, see API Reference.)

To add the SuperQBE templates to your application, you must begin by adding the Support Files and the Global Support extension template. Once you've done this, you can proceed to add one or more of the other templates.

For more information, see:

- Support Files
- Global Extension Template
- QBE Browse Control
- QBE Form Control
- Optionally Filter Tagged Record in a BrowseBox
- Filter Tagged Records in a BrowseBox
- Apply QBE Filter to Process/Report
- Get Tagged Record in Process/Report
- Filter Tagged Records in a Process or Report

2.2 Support Files

(QBE.TXD and TAGGING.TXD)

In SuperQBE 4.0, we decided to always place the QBE and Tagging support files in the dictionary. It gives us many benefits, and clears up much confusion regarding the settings for the global extension template.

Importing the QBE Files

The first step is to load your dictionary, and import `QBE.TXD` from `SUPER\LIBSRC\QBE`. You'll end up with the following files in your dictionary:

- SQBE::Set** This file contains the names for the saved criteria sets. Each record contains the criteria set number (-1 for the last used criteria), source (procedure name), date saved, description, and user (when applicable). It also contains the filter expression, which will be used in future versions of QBE.
- SQBE::Criterion** This file is a "child" of `SQBE::Set`. It contains the individual field criteria in the set. Each record represents one match type. (If a single field has multiple criteria specified, then it will contain multiple records here.)

Importing the Tagging Files

The first step is to load your dictionary, and import `TAGGING.TXD` from `SUPER\LIBSRC\TAGGING`. You'll end up with the following files in your dictionary:

- TagFile_** This file stores tags when you are using "Disk Set" with a file whose primary key field is an integer (i.e.: `LONG`). In addition to the reference to the data record, it also contains the tag set number and user (when applicable).
- TagFilePos_** This file stores tags when you are using "Disk Set" with a file whose primary key field is not an integer (i.e.: `STRING`). In addition to the reference to the data record, it also contains the tag set number and user (when applicable).
- TagSet_** This file contains the names for the saved tag sets. Each record contains the tag set (always a negative number), source file name, count, and user (when applicable).

2.3 Global Extension Template

To use the QBE features, you must add the global support to your application. The *QbeGlobal* extension template can be found within the "Class *SuperQbeABC*".

There are actually no settings for the template. For those of you familiar with old SuperQBE versions, the following settings are in place:

Location of QBE Library - This is determined by your setting of "Generate template globals and ABC's as EXTERNAL". If this setting is ON, then the tagging support library is external. Otherwise, it's internal.

Location of File Definitions - As of SuperQBE 4.0, you must import all support files into your dictionary.

Support Select with Tagging Library - ON

Support tagging using TagFile_ (POINTER) - ON

Support tagging using TagFilePos_ (POSITION) - ON

Omit ST::SelectTagSet Procedure - OFF

Location of Tagging Library - This is determined by your setting of "Generate template globals and ABC's as EXTERNAL". If this setting is ON, then the tagging support library is external. Otherwise, it's internal.

Location of File Definitions - As of SuperQBE 4.0, you must import all support files into your dictionary.

POSITION() Size - 1024

Flag Field Tagged Value - This setting has moved to the individual Control, Extension, and Code templates.

2.4 Browse Control

(Control Template)

Normally you will be performing QBE operations by calling a Form from a Browse. First you have to add the QbeBrowse Control template to your Browse screen. There are only two settings for this template:

Form Procedure - This is the Form procedure that is called to perform the search operation. Normally this will be the same as your Update Procedure.

Show only selected after successful search - This controls how the system behaves after your form has returned to the Browse. If this is OFF, then the browse will do nothing after the QBE Form returns. If it is ON, then the action will depend on your approach:

1. **NEW in Super QBE 6.0:** If your Form was set to return the QBE Filter and exit, then the browse will automatically apply that filter. Be careful, though. *The Browse's file schematic must include all of the files that are referenced by the Form.* Otherwise, your filter may reference fields that are not available in the browse's view structure.

The filter itself is saved by the Form and fetched by the Browse using the name of the primary file, so this must also be the same for both. The actual API calls are:

```

!In the Form:
SQBE::Global.SetFilter('Primary', <FilterExp>)

!In the Browse:
FilterExp = SQBE::Global.GetFilter('Primary')
    
```

2. If this option is on, then the template will attempt to find a "BrowseOptTagFilter" Control template or "?OnlyTagsTab" tab to display only tagged records. Otherwise, all records will be shown.
3. If your window has a tab with an equate of "?OnlyTagsTab" and it does not have a BrowseOptTagFilter button, then the QBE template will automatically switch to that tab after a successful search. You are responsible for writing the filter expression using the "Validate Record: Filter Checking" embed, with some code like this:

```

IF CHOICE(?CurrentTab) = 2
  IF ~GetTag_(eT_TagSet, Pre:Field)
    RETURN Record:Filtered
  END!IF
END!IF
    
```

In this example, "CHOICE(?CurrentTab) = 2" is the condition for your ?OnlyTagsTab.

"eT_TagSet" is your tag set number. "Pre:Field" is your primary key field. Here are some other options for that line:

```
IF ~GetTag:PtrM(eT_TagSet, Pre:Field)
IF ~GetTagPos_(eT_TagSet, Pre:Field)
IF ~Pre:FlagField
```

Warn of possible wait before refilling the browse? - If your browse usually takes a long time to show the first page of query results, then you may want to warn the user to wait while that takes place.

Offer to clear previous search filter? - Do you want to ask the user whether they wish to clear the previous filter before applying the new filter for the latest query?

Execute code after successful search? - *(NEW in Super QBE 6.11)* If you don't want the browse to filter through all of the matching records, then you may elect to create a special procedure that shows only the matches, in a more efficient or customized manner. You can execute that code here.

Also after QBE Form sets filter? - *(NEW in Super QBE 6.11)* This applies only to the ABC chain, and is related to the prior setting. You can call this code if the QBE Form returns with a calculated filter (based upon your settings in the QBE Form template). To grab the filter, see the necessary assignment in the generated code. Search for `QbeResponse:CalcFilter`.

2.5 Form Control

(Control Template)

This is where the bulk of the query operation is performed. The QbeForm Control template handles the user interface for entering the search criteria, as well as performing the searching operations. There are a large number of settings to control the behavior of the system.

General Tab

Basic Interface: Wizard+Regular

Alternate Update Proc:

Interface Options:

- Offer to "Restore search criteria from last session"
- Use Dictionary Descriptions for fields in Wizard
 - Append Descriptions to Field Names
- Move "Begin Search" button to location of "OK" button

Buttons Unavailable: HIDE

Criteria Window Help ID:

Basic Interface - Specify whether you want your user to utilize the Wizard, the Regular form, or both Wizard+Regular. If you specify Wizard+Regular, the wizard will appear first. Once the user has pressed the Finish button, the form will appear as usual.

"Wizard" Interface:

SuperQBE Wizard

The available database fields are listed below. You can specify search criteria for any or all of these.

Highlight the desired field and press the [Change] button.

- Child.Birthday
- 1 Child.Name
- Customer.Birthday
- Customer.City
- Customer.Comments
- Customer.FName
- Customer.LName
- Customer.MaritalStat
- Customer.No
- Customer.Phone
- Customer.Retired

The currently highlighted field's criteria appear in the list box below.

Press [Next] when you are done entering criteria.

- B. HAN

Step 3 of 5
Enter Criteria

< Back Next > Change Finish Cancel

"Regular" Interface:

Alternate Update Proc - This is used when you want this procedure to perform only searches, while another procedure performs the actual updates to the file. If you specify this setting, it will call the alternate update procedure in two situations:

1. The form is called when GlobalRequest <> QbeRequest.
2. The user chooses to edit a record during the search.

Interface Options:

Offer to "Restore search criteria from last session?" - This is a convenience feature. If your users usually repeat the same query as their last one, or if they tweak it just a bit, then this can save them lots of time.

Use Dictionary Descriptions for fields in Wizard - If you are using the Wizard, this will control whether each field should be represented by its description from the dictionary (if available) or by "Filename.Fieldname".

Append Descriptions to Field Names - Do you want to see both field names and descriptions?

Move 'Begin Search' button to location of 'OK' button - This affects the positioning of the "Begin Search" button. These two buttons are never available at the same time, and each of them is the default when it is available. This option allows you to have them coexist-exist in the same space on the screen at run-time. You don't need to have them in the same place during design, though, which keeps alleviated the need to have one button hidden by the other.

Buttons Unavailable - This controls the appearance of the two QBE buttons during normal editing operations. The default is "HIDE", although you may want to change it to "DISABLE".

Criteria Window Help ID - If you've created a help window for your users to better understand the criteria settings window, then you can place its topic ID here.

Operations Tab

<input checked="" type="checkbox"/> Support "Select" and "Unselect"	
<input type="checkbox"/> Compute QBE Filter and return (without Tagging) A calling browse can automatically use this filter.	
<input checked="" type="checkbox"/> Support "Unselect" from existing results	
<input type="checkbox"/> Automatically clear old tags (when not Unselecting)	
<hr/>	
<input checked="" type="checkbox"/> Support "Edit"	
<input type="checkbox"/> Place [Edit] before [Select]+[Unselect] buttons	

Support "Select" (and "Unselect") - This controls whether the user is given the opportunity to select and (sometimes) unselect records. This is ON, then the Tag Storage tab will be available. This is the only method compatible with the Wizard-only interface.

Compute Filter, then return without Tagging (*NEW in Super QBE 6.0*) - The "classic" approach of Super QBE is to search for records using the Form, tag all the matching records, and then return to the caller. If your caller is a Browse, then it would have to filter using these tags. This two step process could be rather slow (especially if you're using a SQL back-end).

Now you have the option to compute the filter expression in the Form and then exit. At that point, the Browse can use this filter expression to display the results much quicker. The calling browse will automatically use the filter expression (assuming that it's using the QBE control template to call the form.)

NOTE: This circumvents the entire tagging process, so you will not be able to use additional queries to unselect from the result set, or to use the tag filters on your reports. Instead, will have to use the "QBEPProcessFilter" template on your processes and reports.

Support "Unselect" from existing results - You may want to support Selecting records, but you don't want to allow the users to perform additional queries to unselect from the previously selected results.

Automatically clear old tags (when not Unselecting) - As a default, the user will be asked whether to clear existing tags before starting a new selection. If you want to suppress this verification, so that tags are automatically cleared, then turn this on.

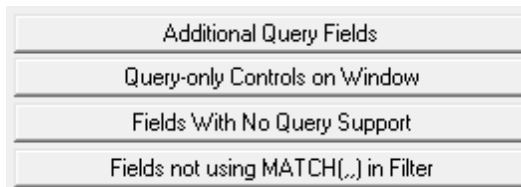
Support "Edit" - This option is only available if the Basic Interface is not set to "Wizard". It controls whether user will be given the option of "Searching for records to Edit". It will preview each matching record, asking whether the user wants to edit it, or to proceed to the next record.

Place [Edit] before [Select]+[Unselect] buttons - This controls whether the Edit button will appear before Select and (sometimes) Unselect buttons on the search task window.

Tag Storage Tab

See "Tag Storage Options" in the Introduction.

Fields/Controls Tab



Additional Query Fields in Wizard

This button leads to a list of extra fields that will be included in the query. Normally, the user can access only fields that are present in the form. If you are using the Wizard, this will add those additional fields to the Wizard interface. The settings (e.g.: entry picture) are determined by the dictionary.

Query-only Controls on Window

This button leads to a list of controls that are used only during the query process. They will be hidden when the form is used for regular update operations. This is very handy for fields from related files that have nothing to do with the update form.

NOTE: Group the fields from related files onto one or more tabs, then declare the tabs as query-only fields. This will hide all controls on each of the tabs.

Fields with No Query Support

This button leads to a list of controls that are used only when editing data. When in QBE mode, they are automatically hidden.

Fields not using MATCH(,,) in Filter *(NEW in Super QBE 6.0)*

Super QBE 6.0 introduced the MATCH function in filter expressions. It's applicable primarily for STRINGS, when performing an "Equals", "Not Equals", "Begins With", "Ends With", "Contains", and "Doesn't Contain". It's also used for numeric picture fields, like phone numbers, for "Begins With" and "Ends With".

There are some database backend fields that are incompatible with MATCH. For example, in MS SQL Server, using MATCH is acceptable for most numeric fields, because they are automatically converted to strings for the comparison. However, MONEY fields are not.

We've also received a report that MATCH can slow down searches with TopSeed files on a network, although we've not confirmed this ourselves yet.

(NEW in Super QBE 6.03) - You can disable the use of MATCH(,,) entirely with single setting, rather than needing it to be done for each field individually.

In most situations, you shouldn't need to include anything in this list. If you run into a problem, though, you can prevent usage of MATCH on a column-by-column basis. For those fields, it will revert to the old filter expressions. Most of the time these will still be passed through the driver to the backend database, for fast processing. Occasionally, however, the system will resort to client-side filters, which can be quite slow for large databases.

Defaults Tab

Default Match Types:	
String:	Begins with
Number:	Equals
Picture:	Equals
Date:	Equals
Option:	Equals
Check:	Equals
Text:	Contains

This tab leads to the default settings for the different data types. For example, you can individually control the default comparison type for String, Number, Picture, Date, Check, Option, and Text fields.

Events Tab

Normally your own event handling will not be processed during criteria entry.

If you want to your own code processed during this time, then turn ON the switch below.

When necessary, check the mode using:

IF ThisWindow.Request = QbeRequest

Pass events through during QBE

Except for "Hidden QBE Criteria", all of the QBE-related embed points have been removed in version 4.0. Instead you can use the regular embed points offered by Clarion. By default, however, the templates intercept all events processed by the TakeEvent method, so that your code will not be receive them during QBE criteria entry.

If you want events to be passed through to your code during QBE mode, then turn this setting ON. Be forewarned, though, that you are responsible for check the status so that your embeds that are update-related are not processed during QBE operations. This can be done by check if `ThisWindow.Request=QbeRequest`.

Not all events will be passed through. If an event is used by the QBE code itself, then your code will never see it.

Misc. Tab

Last Sort Character

If your back-end database uses a non-ASCII sorting sequence, you should change this to something else like 'Z'.

Last Sort Character:

Last Sort Character - Most of the time, the default tilde (~) is acceptable. However, many backends (e.g. many SQL systems) don't use the same ASCII sorting sequence. In this case, you'll need to change this to z, or some other character. Check the documentation for your database to determine the

last character in its sorting sequence.

Embeds

There is only one event provided for your use. (The others were removed in version 4.0. See the "Events Tab" description above.)

Hidden QBE Criteria - In some situations, your QBE operation must search within a standard set of records. For example, you might restrict the searches to transactions from this year, or to records assigned to the current user. You don't want to force the user to enter this criteria each time. In the case of user-specific records, there may be security issues involved. This embed point is designed to handle this problem. You can add as many hidden criteria as you need. The code will look something like this:

```
IF ~Condition THEN RETURN Record:Filtered.  
  
IF YEAR(Inv:Date) <> YEAR(TODAY()) THEN RETURN Record:Filtered.  
  
IF Cus:UserNo <> UserNo_ THEN RETURN Record:Filtered.
```

2.6 Optionally Filter Tagged Records in a BrowseBox

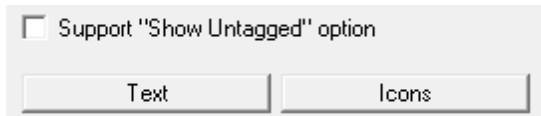
(Control Template)

Your users may wish to switch between viewing "all records", "only tagged records" and "only untagged record". The "BrowseOptTagFilter" Control template is designed with this goal in mind. It is populated onto the browse window as a button that changes its text and icon depending on the current viewing state. The extension settings are as follows:

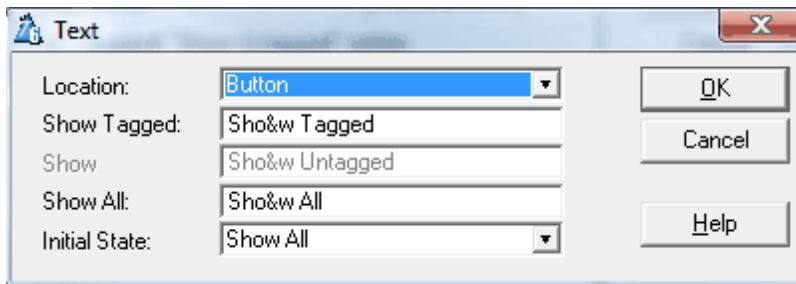
Tag Storage Tab

See the "Tag Storage Options" in the Introduction.

Button Text Tab



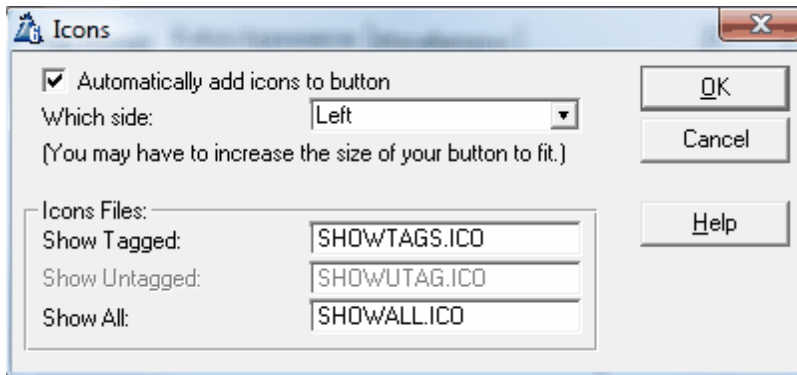
Support "Show Untagged" Option - This enables the "Show Untagged" state.



Text Button - These are the phrases that will be displayed on the button. You can choose to place the text on the button, or in the tool tip (in case the button is large enough for only an icon).

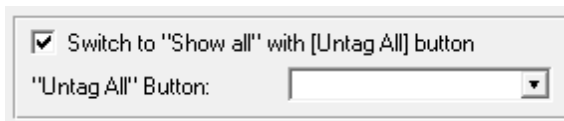
The "Show Tagged" message is displayed in the button when they are viewing all records. By pressing the button, it will change to the "Show Untagged" or "Show All" text (depending on other settings), and they will be in another filter state. In each of these text settings, you can include an ampersand (&) to control the "hot key" for the button.

The final option is a drop box controlling the initial state when the user enters the browse. The default for this is "Show All".



Icons Button - If desired, you can have icons automatically displayed in the button. You can control whether they are displayed on the left, right or default (center). You also specify the icon files to be used.

Miscellaneous Tab



Switch to "Show All" with [Untag All] Button (NEW in Super QBE 6.03) - In most cases, this template will be used alongside the tagging buttons control template. In those situations, it's helpful to have the filter mode reset to "Show All" if the [Untag All] button is pressed. Otherwise, the browse window will be empty, after the user waits for all records to be read and discarded.

In case you have more than one set of BrowseTaggingButtons, you must also specify the name of the "Untag All" control.

2.7 Filter Tagged Records in a BrowseBox

(Extension Template)

This Extension template is similar to the "BrowseOptTagFilter" control template. It is used to display only tagged records on a Browse. The only difference is that this template has a fixed filter, rather than giving the user the option of toggle between filtered and non-filtered. This extension can be used only with a procedure already containing a BrowseBox.

Tag Storage Tab

See the "Tag Storage Options" in the Introduction.

If your browse used "BYTE Field in Primary" as its tag storage method, then you don't need to use this extension. Instead, you could use a key on the tag field to make the Browse very fast. This is not really necessary unless you have many records in your file. We'll leave that up to your personal preference.

2.8 Apply QBE Filter to Process/Report

(Extension Template)

This template applies the filter expression from the most recent search to this Process/Report.

Optional filter (if filter is available)

Filter Question:

This extension template is used when you are running a Report or Process after a QBE Form has stored a filter expression. It will automatically apply that filter expression and apply it to the process or report.

Be careful, though. *The Process or Report's file schematic must include all of the files that are referenced by the Form.* Otherwise, your filter may reference fields that are not available in this procedure's view structure.

The filter itself is saved by the Form and fetched by the Process/Report using the name of the primary file, so this must also be the same for both. The actual API calls are:

```
!In the Form:
SQBE::Global.SetFilter('Primary', <FilterExp>)
```

```
!In the Process/Report:
FilterExp = SQBE::Global.GetFilter('Primary')
```

This filter will be *in addition* to your any pre-programmed filters for that procedure.

Optional filter (if filter is available) - In some cases you may want to give the user a choice of whether to process all records, or those matching the filter.

If there is no filter expression computed and stored by a QBE Form, then the question will not be asked, and the Process/Report will automatically include all records.

Filter Question - If the previous setting is ON, then enter the question to be posed to the user.

2.9 Get Tagged Data Records in a Process/Report

(Extension Template)

This extension template is used when you are running a Report or Process directly against TagFile_ or TagFilePos_. It automatically retrieves the tagged record based upon the saved Ptr/Pos value. The reports and processes are very fast, because only the tagged records are retrieved. The template automatically sets the filter for the view. If your data file is related to your tag file, you can also specify the processing order.

To use the Order feature, your data file must be related to your TagFile. Set up a relationship between either TagFile_ (if your primary key field is an integer) or TagFilePos_ (if your primary key field is a string). The data file will be on the "ONE" side, with the TagFile on the "MANY" side. Do *not* specify a foreign key for the TagFile. The data file will use its primary key. Equate the data file's primary key field with the Ptr/Pos field from TagFile.

The primary file for the Process or Report will be TagFile_ or TagFilePos_. Do **not** specify a key. Add the data file as a secondary file to the TagFile. This file must be the same as the original file with the tagged records.

Next you need to add the ProcessGetTaggedData extension template. Once you've done this, you will see the following settings:

Tag Storage Tab

See the "Tag Storage Options" in the Introduction.

You must also enter the data file to access. This will be your datafile (which has been specified as secondary to the TagFile in the file schematic.)

Order Tab

You can specify the processing order here. Enter the clause manually, or have the template assist you. If you choose to do it manually, the order clause will take the following form:

```
+Cus:LastName, +Cus:FirstName, -Cus:Birthday
```

As an alternative, you can specify a field name so that the order is determined by your program logic at run-time. To do this, prefix the field name with an exclamation point (!):

```
!Loc:Order
```

If you would rather use assisted entry, then you can specify each field component and its order by picking it from the fields list. Make sure you include only fields that are part of the TagFile file schematic.

Miscellaneous Tab

Perform Secondary Fetches for SQL - The REGET command sometimes doesn't seem to handle secondary files properly, especially with SQL files. For SQL, or if the process doesn't seem to be updating the records properly, try turning this switch ON.

2.10 Filter Tagged Records in a Process/Report

(Extension Template)

Once your users have tagged records, they will want to do something with them. The most common request is to print only tagged records in a report. Or they may want to delete the tagged records using a Process procedure. You can do this by adding this extension to your procedure.

Tag Storage Tab

See the "Tag Storage Options" in the Introduction.

If your browse used "BYTE Field in Primary" as its tag storage method, then you don't need to use this extension. Instead, you could use a key on the tag field to make the Browse very fast. This is not really necessary unless you have many records in your file. We'll leave that up to your personal preference.

Filter Settings Tab

If No Tags Exist	
<input checked="" type="checkbox"/>	Process all records if no tags exist
<input type="checkbox"/>	Abort procedure if no tags exist
"Aborting" Message:	<input type="text" value="No tags found! Aborting..."/>
(Leave blank for no message window.)	
If Tags Exist	
<input checked="" type="checkbox"/>	Optional filter if tags exist
Filter Question:	<input type="text" value="Show only tagged records"/>

If No Tags Exist:

Process all records if no tags exist - If there are no tags when the report/process is called, then it automatically processes all records if this is turned ON. If you turn this OFF, then no records will be processed.

Abort procedure if no tags exist - If the previous setting is OFF, then this option controls whether the procedure should immediately abort, or to continue (even though no records will be processed).

"Aborting" Message - If the Abort setting is ON, then you can specify an optional message here.

If Tags Exist:

Optional filter if tags exist - In some cases you may want to give the user a choice of whether to process all records, or only tagged records.

Filter Question - If the previous setting is ON, then enter the question to be posed to the user.

Miscellaneous Tab

Wipe tags after completion - *Do you want the tags to be cleared when the report/process is complete?*

3 Appendices

3.1 Printing Tagged Records with ReportWriter

If you are using Report Writer, there is a good chance that you would like to process tagged records. This is not too difficult. You have two options. If your tags are stored in a BYTE Field in your data file, you can simply use a filter. Of course this not support multi-user access, as everyone is using the same tag field. If there is normally only one person working with tags, however, then this may be good enough.

In most situations, though, you'll have to use one of the other tagging methods. You should start by understanding the description of the ProcessGetTaggedData extension template, because you will be using much the same method.

If you are using Memory tags, then you will have to use the CopyTags code template to copy them to the corresponding TagFile storage.

The TagFile must be defined in your dictionary. To bring the tag files into your dictionary, import TAGGING.TXD. Your data file must be related to the TagFile, with the data file on the "ONE" side of a "1:MANY" relationship. Once this is done, your report will process the TagFile as the primary file, with your data file as a related file.

Then you must specify a filter so that only corresponding tags are processed. If only one user is tagging records and you are only tagging in a single file in your entire APP, then you could theoretically skip this step. Otherwise, you must set up a filter something like this:

```
TF_:Usr=UserNo AND TF_:Tb1=TagSet
```

"UserNo" is the current user number. If you are not using Super Security, then this will always be zero (in which case you can omit that portion of the filter expression). Otherwise, it will be stored in a variable called UserNo_. You must pass the user number from the program to ReportWriter, or you must have the user enter it in a run-time field.

The TagSet number must match the original number. (If you are using equates, then you must use the literal number in the report filter.)

For example, if you are not using Super Security, and your TagSet is 1, the filter would be:

```
TF_:Usr=0 AND TF_:Tb1=1
```

Finally, you can change the order of the output to match your needs.

3.2 Multi-APP Development using LIBs/DLLs

If you are using the QBE templates and you want to split your application into multiple LIBs/DLLs and a single EXE, here's how you do it. This description is based upon the example application in SUPER\EXAMPLES\QBE2.

LIB/DLL with the Support Code

1. If you haven't already done so, import the QBE and Tagging support file definitions from SUPER\LIBSRC\QBE\QBE.TXD and TAGGING.TXD.
2. Create or open the APP destined to be a DLL/LIB. This must be your base APP in which all of the ABC support code is included.
3. Choose "Application / Properties ..." from the pulldown menu.
4. Blank the "First Procedure" field.
5. Ensure that your "Destination Type" is set to "Dynamic Link Library (.DLL)" or "Library (.LIB)".
6. Click [OK].
7. Press the [Global] button.
8. Press the [Extensions] button.
9. Press the [Insert] button.
10. Select "QbeGlobal" under "Class SuperQbeABC".
11. Press [OK] to exit the "Extensions and Control Templates" window.
12. Turn OFF the "Generate template globals and ABC's as EXTERNAL" check box.
13. Press [OK] to exit the "Global Properties" window.
14. Make the DLL/LIB. (Press the "Lightning" on the button bar, select "Project / Make" from the pulldown, or press Ctrl-M.) You'll find a LIB file in your application directory.
15. If you specified your destination type to "DLL" back in step #5, then you will also have a DLL in your current directory. Remember to include this file when you distribute your APP.

EXE/LIB/DLL without the Support Code

1. Create or open the APP that will call the procedures in the APP described above.
2. Press the [Global] button.
3. Press the [Extensions] button.
4. Press the [Insert] button.

5. Select "QbeGlobal" under "Class SuperQbeABC".
6. Press [OK] to exit the "Extensions and Control Templates" window.
7. Turn ON the "Generate template globals and ABC's as EXTERNAL" check box.
8. Press [OK] to exit the "Global Properties" window.
9. Select "Application / Insert Module", then choose "ExternalDLL".
10. Specify your base application name with a .LIB extension as the Name (e.g.: BASE_APP.LIB), then press [OK] to exit the Module Properties window. (Even if your base APP is a DLL, you still use a .LIB extension here.)
11. Make the APP. (Press the "Lightning" on the button bar, select "Project / Make" from the pulldown, or press [Ctrl-M].)
12. Test your APP.

Remember, if you are using DLLs, then all references must be down. For example, you are three APPs: AAA, BBB and CCC. BBB and CCC are DLLs, while AAA is an EXE. AAA calls routines in BBB and CCC. BBB calls routines in CCC. Therefore, CCC must be made first, then BBB, then AAA. If BBB did not call anything in CCC, then you could make BBB before CCC, if desired.

Libraries are different. You can have as many cross references as you wish. LIBs can even call routines in the parent EXE. This is because the linker has all modules (EXE and LIB) together at once, so it can rectify all references, none of which are resolved in a LIB. This contrasts a DLL, which is a standalone module with all of its external references resolved.

3.3 Interface Modification and Translation

We've moved all displayable strings to StAbQbe.trn and StAbTag.trn. These files contain equates for messages, titles, buttons and prompts throughout the Super QBE system. You'll also find various window definitions here (including the Wizard).

Feel free to change any or all of these. In most cases, you should copy the file into your application directory. That way new versions of the templates don't overwrite your preferences. For each new APP you create, you can copy your version of the file into that directory.

The system also uses various icons. The icon names are hard coded into the templates, but you are welcome to create your own version of the icons using the same names. Again, ensure that they are found in your RED search path before the standard files.

SQBE.ICO	Search button on Browse
SQBE_HLP.ICO	Search Help button on Form
SQBE_GO.ICO	Begin Search button on Form
SQBE_IG.ICO	Ignore this field
SQBE_EQ.ICO	Equals
SQBE_NEQ.ICO	Doesn't Equal
SQBE_GT.ICO	More Than
SQBE_LT.ICO	Less Than
SQBE_GE.ICO	More or Equal
SQBE_LE.ICO	Less or Equal
SQBE_IR.ICO	Inside Range
SQBE_OR.ICO	Outside Range
SQBE_BW.ICO	Begins With
SQBE_EW.ICO	Ends With
SQBE_CO.ICO	Contains
SQBE_NCO.ICO	Doesn't Contain
SQBE_EM.ICO	Month
SQBE_ED.ICO	Day
SQBE_EY.ICO	Year
SQBE_EMD.ICO	Month+Day
SQBE_EYM.ICO	Year+Month
SQBE_MV.ICO	Multi-Criteria
SQBE_WIZ.WMF	Wizard Portrait
SQBE_WZI.ICO	Wizard Introduction
SQBE_WZL.ICO	Wizard Load
SQBE_WZS.ICO	Wizard Save
SQBE_WZF.ICO	Wizard Finish

3.4 Example Programs

There are four example programs provided with the Super QBE templates: two each for ABC and Clarion/Legacy. See the Installation topic for their location.

The only difference between QBE1 and QBE2 is that the second demonstrates how to place the tagging support module into a DLL. In the #2 directory, you'll find SUPPORT.APP, which must be built before TEST.APP.

NEW in Super QBE 6.0: The key difference between the ABC and Clarion/Legacy APPs, is that the ABC examples demonstrate the new QBE Filter feature.

Each of the examples includes a basic browse and form used in a typical QBE operation. There are a wide variety of control types on the form to demonstrate the different match type windows that will appear. Try entering a few different criteria and see what comes out. If you enter "begins with 'H'", it will use the last name key to quickly find all names starting with that letter.

After a successful search, the browse will display only matching (tagged) records. Or in the case of the new QBE Filter feature, the browse uses the filter computed by the form.

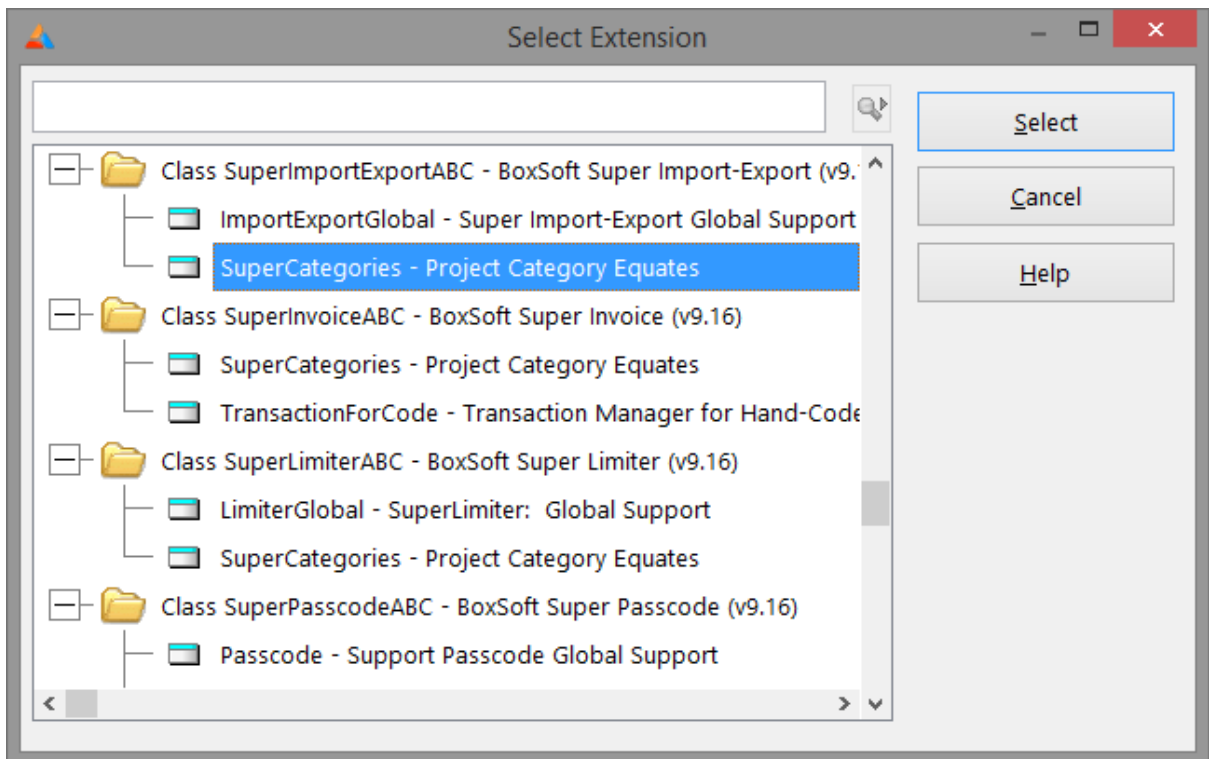
There are also reports accompanying each of these.

3.5 Project Defines

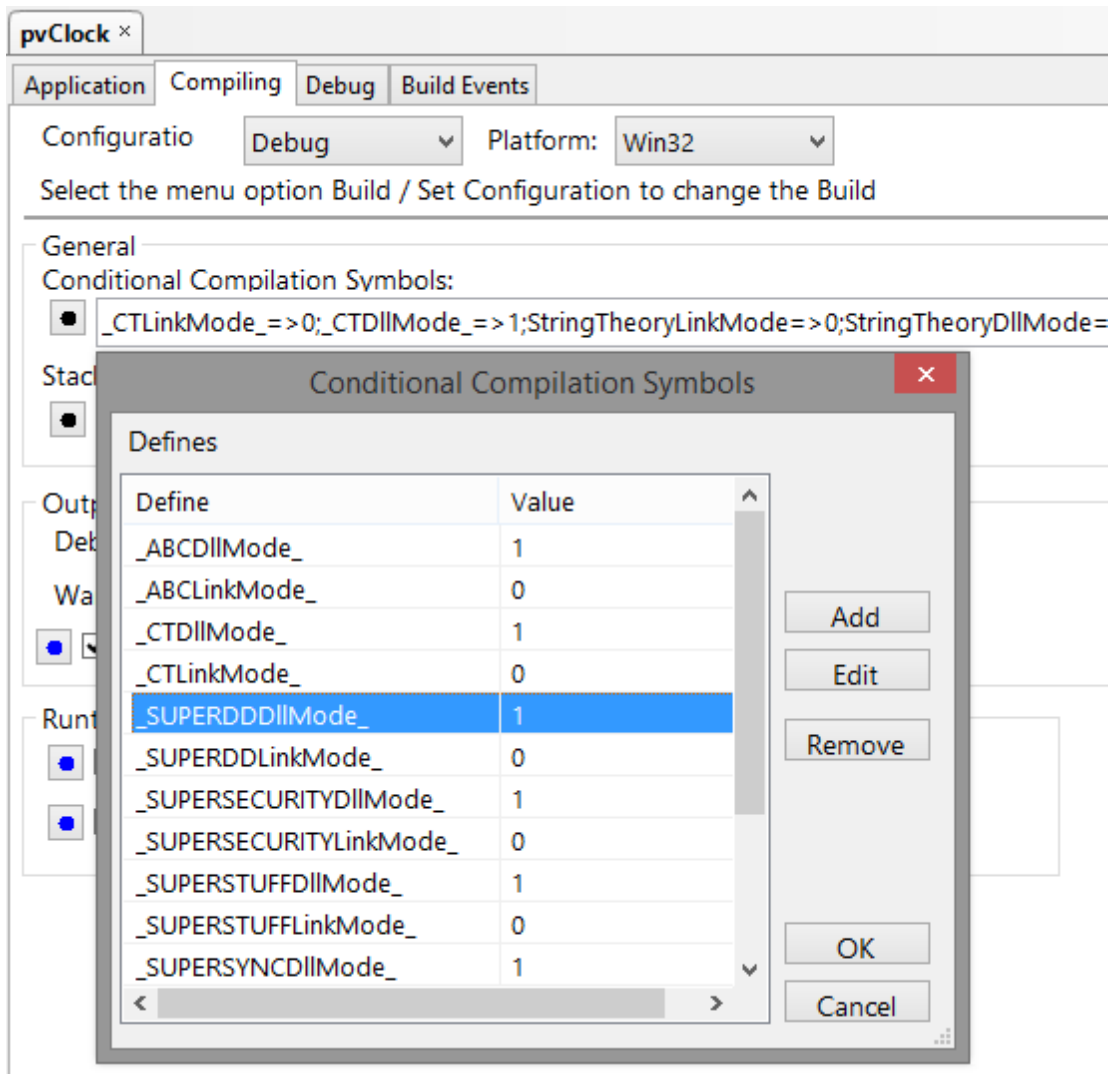
Prior to our 7.0 template versions, we were utilizing the same LINK and DLL Project Defines as the ABC libraries: `_ABCLinkMode_` and `_ABCDIIMode_`. This caused all of our libraries to be included in your base/dictionary DLL, even if you weren't using them in a particular development project.

To make this inclusion more selective, as of our 7.0 versions we changed to use various other switches. Some of these are product related (e.g. Super QBE uses `_SuperQBELinkMode_` and `_SuperQBEDIIMode_`), while others are associated with one of our shared base classes (e.g. Drag & Drop uses `_SuperDDLLinkMode_` and `_SuperDDDIIIMode_`). Usually the templates (especially the global ones) automatically add the necessary entries to your Project Defines. If you happen to use the templates in your APPs in the wrong combination, these can be inadvertently omitted.

For APP-based systems, you can force the switches to be included by using the Super Categories global extension template. Every one of the Super Templates has this extension to apply its own switches, so if you're using multiple templates in a particular APP, you may have to add this extension for each of the products. (As was mentioned above, if there's already a global extension populated for a given Super Template, then you don't have to add this extension for that product.) Even if it's not needed, there's no problem with adding the SuperCategories global extension.



For hand-coded PRJ-based systems, you must add the switches manually. Take note of their names in the INC files, and then add them to the project settings like this:



3.6 API Reference - QBE

SQBE::Global Object (*NEW in Super QBE 6.0*)

SQBE::Global.SetFilter (STRING File, <STRING Filter>, BYTE Sync=1)

Sets the stored filter expression for the named file.

If the Filter parameter is omitted, then the filter is cleared. (It's the same as passing an empty string.)

You can pass zero (0) for the Sync parameter, if you've already called SQBE::Global.Sync.Wait. By the way, you can safely call our Sync methods in both Clarion 5.5 and 6. (In the case of C55, the Sync does nothing.)

```
!Filter customers whose names starts with "H":
MyFilter = 'MATCH(Cus:Name,'H*', '& MATCH:Wild+MATCH:NoCase &)'
SQBE::Global.SetFilter('Customer', MyFilter)

!Clear filters for Customer, Invoice and Product:
SQBE::Global.Sync.Wait
SQBE::Global.SetFilter('Customer',, False)
SQBE::Global.SetFilter('Invoice',, False)
SQBE::Global.SetFilter('Product',, False)
SQBE::Global.Sync.Release
```

SQBE::Global.GetFilter (STRING File, BYTE Sync=1), STRING

Returns the stored filter expression for the named file. You can pass zero (0) for the Sync parameter, if you've already called SQBE::Global.Sync.Wait.

```
BRW1.SetFilter(SQBE::Global.GetFilter('Customer'),'Q')
```

3.7 API Reference - Tagging

Tagging API for Memory (Integer Primary Keys)

If you are using the "Memory Set" storage method and some of your data files have integer primary key fields, then the following procedures and functions apply:

ClrTag:PtrM (TagSet, PrimaryKey)

This procedure is used to clear a tag for a particular tag set and primary key.

```
ClrTag:PtrM(1, Cus:No)
```

CntTag:PtrM (TagSet)

This function returns the number of tags in the specified tag set.

FstTag:PtrM (TagSet)

This function returns a pointer to the first tagged record in the specified tag set. If there are no records in the specified tag set, the function returns zero.

There are a number of situations where you would use FstTag:PtrM(). You may want to check if there are any tagged records available. You would also use FstTag:PtrM() in conjunction with NxtTag:PtrM() to process all tagged records.

GetTag:PtrM (TagSet, PrimaryKey)

This function returns either 1 or 0, depending on whether the requested record is tagged.

```
IF GetTag:PtrM(1, Cst:No)
  DO Process
END
```

ListTags:PtrM (TagSet)

This function returns a string with a list of tags as '[1][2][3]...'. This can be useful for filters with INSTRING functions, etc. The maximum string length is 1000 characters. If your tag set exceeds that, an ASSERT will fail.

LstTag:PtrM (TagSet)

This function returns a pointer to the last tagged record in the specified tag set. If there are no records in the specified tag set, the function returns zero.

There are a number of situations where you would use LstTag:PtrM(). You may want to check if there are any tagged records available. You would also use LstTag:PtrM() in conjunction with PrvTag:PtrM() to process all tagged records.

NewTag:PtrM (TagSet)

This procedure clears all tags for a specified tag set.

NxtTag:PtrM (TagSet)

This function returns a pointer to the next tagged record in the specified tag set. `FstTag:PtrM()` must be called first for this function to work. If there are no more records, the function returns zero.

PrvTag:PtrM (TagSet)

This function returns a pointer to the previous tagged record in the specified tag set. `LstTag:PtrM()` must be called first for this function to work. If there are no more records, the function returns zero.

RvsTag:PtrM (TagSet, PrimaryKey)

This procedure is used to reverse a tag for a particular tag set and primary key. That is, if the tag is set before the call to `RvsTag:PtrM` it will be cleared, and vice versa. There is an optional Value parameter, for storing the tags in a particular order for `FstTag:PtrM` / `NxtTag:PtrM` processing.

```
RvsTag:PtrM(1, Prd:No)
RvsTag:PtrM(2, Cus:No)
```

SetTag:PtrM (TagSet, PrimaryKey)

This procedure is used to set a tag for a particular tag set and primary key. There is an optional Value parameter, for storing the tags in a particular order for `FstTag_/NxtTag_` processing. It returns True if newly tagged, or False if already tagged.

```
SetTag:PtrM(1, Prd:No)
SetTag:PtrM(2, Cus:No)
```

ShutDownTag:PtrM ()

This procedure clears all of the tag queues in memory before the program exits.

Tagging API for Memory (Non-Integer Primary Keys)

If you are using the "Memory Set" storage method and some of your data files have non-integer primary key fields, then the following procedures and functions apply:

ClrTag:PosM (TagSet, PrimaryKey)

This procedure is used to clear a tag for a particular tag set and primary key.

```
ClrTag:PosM(1, Cus:ID)
```

CntTag:PosM (TagSet)

This function returns the number of tags in the specified tag set.

FstTag:PosM (TagSet)

This function returns a string pointing to the first tagged record in the specified tag set. If there are no records in the specified tag set, the function returns an empty string ("").

There are a number of situations where you would use `FstTag:PosM()`. You may want to check if there are any tagged records available. You would also use `FstTag:PosM()` in conjunction with `NxtTag:PosM()` to process all tagged records.

GetTag:PosM (TagSet, PrimaryKey)

This function returns either 1 or 0, depending on whether the requested record is tagged.

```
IF GetTag:PosM(1, Cus:ID)
DO Process
END
```

ListTags:PosM (TagSet)

This function returns a string with a list of tags as '[1][2][3]...'. This can be useful for filters with INSTRING functions, etc. The maximum string length is 1000 characters. If your tag set exceeds that, an ASSERT will fail.

LstTag:PosM (TagSet)

This function returns a string pointing to the last tagged record in the specified tag set. If there are no records in the specified tag set, the function returns an empty string (").

There are a number of situations where you would use LstTag:PosM(). You may want to check if there are any tagged records available. You would also use LstTag:PosM() in conjunction with PrvTag:PosM() to process all tagged records

NewTag:PosM (TagSet)

This procedure clears all tags for a specified tag set.

NxtTag:PosM (TagSet)

This function returns a string pointing to the next tagged record in the specified tag set. FstTag:PosM() must be called first for this function to work. If there are no more records, the function returns an empty string (").

PrvTag:PosM (TagSet)

This function returns a string pointing to the next tagged record in the specified tag set. LstTag:PosM() must be called first for this function to work. If there are no more records, the function returns an empty string (").

RvsTag:PosM (TagSet, PrimaryKey)

This procedure is used to reverse a tag for a particular tag set and primary key. That is, if the tag is set before the call to RvsTag:PosM it will be cleared, and vice versa. There is an optional Value parameter, for storing the tags in a particular order for FstTag:PosM / NxtTag:PosM processing.

```
RvsTag:PosM(1, Prd:ID)
RvsTag:PosM(2, Cus:ID)
```

SetTag:PosM (TagSet, PrimaryKey)

This procedure is used to set a tag for a particular tag set and primary key. There is an optional Value parameter, for storing the tags in a particular order for FstTagPos_/NxtTagPos_ processing. It returns True if newly tagged, or False if already tagged.

```
SetTag:PosM(1, Prd:ID)
SetTag:PosM(2, Cus:ID)
```

ShutDownTag:PosM ()

This procedure clears all of the tag queues in memory before the program exits.

Tagging API for TagFile (Integer Primary Keys)

If you are using the "Disk Set" storage method and some of your data files have integer primary key fields, then the following procedures and functions apply:

ClrTag_ (TagSet, PrimaryKey, <Value>)

This procedure is used to clear a tag for a particular tag set and primary key.

```
ClrTag_(1, Cus:No)
```

CntTag_ (TagSet)

This function returns the number of tags in the specified tag set.

FstTag_ (TagSet)

This function returns a pointer to the first tagged record in the specified tag set. If there are no records in the specified tag set, the function returns zero. It processes the tags in TF_:ValKey order, which is usually the order of the original browse or order that the records were tagged.

There are a number of situations where you would use FstTag_(). You may want to check if there are any tagged records available. You would also use FstTag_() in conjunction with NxtTag_() to process all tagged records.

GetTag_ (TagSet, PrimaryKey)

This function returns either 1 or 0, depending on whether the requested record is tagged.

```
IF GetTag_(1, Cus:No)
  DO Process
END
```

ListTags_ (TagSet)

This function returns a string with a list of tags as '[1][2][3]...'. This can be useful for filters with INSTRING functions, etc. The maximum string length is 1000 characters. If your tag set exceeds that, an ASSERT will fail.

LstTag_ (TagSet)

This function returns a pointer to the last tagged record in the specified tag set. If there are no records in the specified tag set, the function returns zero. It processes the tags in reverse TF_:ValKey order

There are a number of situations where you would use LstTag_(). You may want to check if there are any tagged records available. You would also use LstTag_() in conjunction with PrvTag_() to process all tagged records.

NewTag_ (TagSet, <Stream>)

This procedure clears all tags for a specified tag set.

NxtTag_ (TagSet)

This function returns a pointer to the next tagged record in the specified tag set. FstTag_() must be called first for this function to work. If there are no more records, the function returns zero.

PrvTag_ (TagSet)

This function returns a pointer to the previous tagged record in the specified tag set. LstTag_() must be called first for this function to work. If there are no more records, the function returns zero.

RvsTag_ (TagSet, PrimaryKey, <Value>)

This procedure is used to reverse a tag for a particular tag set and primary key. That is, if the tag is set before the call to RvsTag_ it will be cleared, and vice versa. There is an optional Value parameter, for storing the tags in a particular order for FstTag_/NxtTag_ processing.

```
RvsTag_(1, Prd:No)
RvsTag_(2, Cus:No)
```

SetTag_ (TagSet, PrimaryKey, <Value>)

This procedure is used to set a tag for a particular tag set and primary key. There is an optional Value parameter, for storing the tags in a particular order for FstTag_/NxtTag_ processing. It returns True if newly tagged, or False if already tagged.

```
SetTag_(1, Prd:No)
SetTag_(2, Cus:No)
```

Tagging API for TagFilePos_ (Non-Integer Primary Keys)

If you are using the "Disk Set" storage method and some of your data files have non-integer primary key fields, then the following procedures and functions apply:

ClrTagPos_ (TagSet, PrimaryKey, <Value>)

This procedure is used to clear a tag for a particular tag set and primary key.

```
ClrTagPos_(1, Cus:ID)
ClrTagPos_(1, Cst:ID)
```

CntTagPos_ (TagSet)

This function returns the number of tags in the specified tag set.

FstTagPos_ (TagSet)

This function returns a string pointing to the first tagged record in the specified tag set. If there are no records in the specified tag set, the function returns an empty string (""). It processes the tags in TP_:ValKey order, which is usually the order of the original browse or order that the records were tagged.

There are a number of situations where you would use FstTagPos_(). You may want to check if

there are any tagged records available. You would also use `FstTagPos_()` in conjunction with `NxtTagPos_()` to process all tagged records.

GetTagPos_ (TagSet, PrimaryKey)

This function returns either 1 or 0, depending on whether the requested record is tagged.

```
IF GetTagPos_(1, Cus:ID)
DO Process
END
```

ListTagsPos_ (TagSet)

This function returns a string with a list of tags as '[1][2][3]...'. This can be useful for filters with `INSTRING` functions, etc. The maximum string length is 1000 characters. If your tag set exceeds that, an `ASSERT` will fail.

LstTagPos_ (TagSet)

This function returns a string pointing to the last tagged record in the specified tag set. If there are no records in the specified tag set, the function returns an empty string ("). It processes the tags in reverse `TP_:ValKey` order

There are a number of situations where you would use `LstTagPos_()`. You may want to check if there are any tagged records available. You would also use `LstTagPos_()` in conjunction with `PrvTagPos_()` to process all tagged records

NewTagPos_ (TagSet, <Stream>)

This procedure clears all tags for a specified tag set.

NxtTagPos_ (TagSet)

This function returns a string pointing to the next tagged record in the specified tag set. `FstTagPos_()` must be called first for this function to work. If there are no more records, the function returns an empty string (").

PrvTagPos_ (TagSet)

This function returns a string pointing to the next tagged record in the specified tag set. `LstTagPos_()` must be called first for this function to work. If there are no more records, the function returns an empty string (").

RvsTagPos_ (TagSet, PrimaryKey, <Value>)

This procedure is used to reverse a tag for a particular tag set and primary key. That is, if the tag is set before the call to `RvsTagPos_` it will be cleared, and vice versa. There is an optional `Value` parameter, for storing the tags in a particular order for `FstTagPos_/NxtTagPos_` processing.

```
RvsTagPos_(1, Prd:ID)
RvsTagPos_(2, Cus:ID)
```

SetTagPos_ (TagSet, PrimaryKey, <Value>)

This procedure is used to set a tag for a particular tag set and primary key. There is an optional

Value parameter, for storing the tags in a particular order for FstTagPos_/NxtTagPos_ processing. It returns True if newly tagged, or False if already tagged.

```
SetTagPos_(1, Prd:ID)  
SetTagPos_(2, Cus:ID)
```

Tagging API Miscellaneous

ST::GetTagUsr ()

This function returns the value of the variable passed into UsrTag_(), or zero if UsrTag_() has not yet been called.

ST::SelectTagSet (Task, Source)

This procedure presents a browse of tag sets for the SaveTags and LoadTags control templates. The user can insert, change, delete and select tag sets. As with other Browse procedure, it sets GlobalResponse appropriately.

The "Task" parameter is either 'S' for Save or 'L' for Load. The "Source" parameter indicates the origin of the tags. This will contain the label of the data file (e.g.: Customer, Employee)

If you don't like the way this window looks, you can turn off this procedure in the global extension options and create your own using the TagSet_ file.

You can import SUPER\LIBSRC\TAGGING\TAGGING.TXA as a starting point. The two procedures use the regular Clarion Browse and Form procedure templates. Make sure that you have the Tag files in your dictionary before you import the TXA file. (Import SUPER\LIBSRC\TAGGING\TAGGING.TXD into your dictionary.)

UsrTag_ (UserNo_)

This procedure is used to set the value of the current "user". If you're using the BoxSoft Super Security templates, then this is handled for you automatically. If you're using another method for security, you need to call this procedure whenever your user changes. Usually this will be only at the start of the program.

3.8 Troubleshooting

Problem

When the form is in search mode, some of your prompts are disappearing.

Solution

Set the control order in the window so that each of your prompts is before its corresponding field in the window structure.

Problem

When you run your application, the system is reporting that the TagFile is bad.

Solution

Because the TagFile is usually a scratch file (unless you are using it to store long term tag selections), just delete and let the system recreate it. The default names for these files are TAGFILE_.TPS and TAGFILEP.TPS.

Problem

The system complains that it cannot find TagSet_ and fields with the prefix TS_.

Solution

Check that you have TagFile_, TagFilePos_ and TagSet_ in your dictionary. If necessary, import them from SUPER\LIBSRC\TAGGING\TAGGING.TXD.

3.9 Contacting Technical Support

If you have any troubles with this product, then please contact:

Mitten Software
2354 West Wayzata Blvd
Second Floor, Suite H
Long Lake, MN 55356

Voice: (952) 745-4941
Fax: (952) 745-4944

Internet: www.mittensoftware.com
answers@mittensoftware.com
www.boxsoft.net
www.boxsoft.net/contact.htm

3.10 License Agreement

One License per Developer

This Super Template product is comprised of the templates, default applications, libraries, source code, documentation, and help files provided with the package. You must have a separate registered copy for each developer using it.

Redistribution

You are allowed to use the product for any programs that you create, and you are permitted to distribute the generated source code. You may not, however, distribute any portion of the product in its original or modified form without the prior written consent from BoxSoft Development.

One exception to this is the example programs provided with this installation or separately from BoxSoft or its agents: these may be distributed without penalty, in either their original or a modified state.

Disclaimer

BoxSoft Development does not warranty this software for any use. Any expenses or lost time due to errors in this product are not the responsibility of BoxSoft Development. We will attempt to fix any errors that are brought to our attention, but we are not legally liable for any lack of correctness of the product.

Index

- A -

Adding QBE to your Applications 20
API Reference 44, 45

- B -

Browse 31, 33
Browse Control 23
BYTE Field in Primary 13

- C -

Class Libraries 6, 42
Contacting Technical Support 53
Conversion 17

- D -

Defines 42
Demos 41
Dictionary 21
Directories 6
Disappear 42
Disk Set 13, 17
DLL 38
DLLMode 42

- E -

Embeds 25
Events 25
Examples 41
External 38

- F -

Filter Tagged Records 23
Filter Tagged Records in a BrowseBox 33
Filter Tagged Records in a Process or Report 36
Form Control 25
Freeze 42

Function Reference 44, 45

- G -

Get Tagged Data Records in a Process 34, 35
Global Extension Template 22
Global Objects 44
GPF 42

- I -

Icons 40
Import 21
Include Files 6
Installation 6
Interface 25, 40
Interface Modification and Translation 40
Internal 38
Introduction 3

- L -

LIB 38
Library Reference 44, 45
License Agreement 54
LinkMode 42

- M -

Memory Set 13, 17
Multi-APP Development using LIBs/DLLs 38

- O -

Optionally Filter Tagged Records in a BrowseBox 31

- P -

Printing 37
Printing Tagged Records with ReportWriter 37
Procedure Reference 44, 45
Process 36
Process:Report 34, 35
Project Defines 42

- Q -

QBE 44

- R -

Redirection File 6
Registering the Template 6
Report 36, 37
Report Writer 37
RTFM Warning 5

- S -

Samples 41
SSEC::Criterion 21
SSEC::Set 21
Strings 40
Support 53
Support Files 21

- T -

Tag Storage 25
Tag Storage Options 13
TagFile_ 21
TagFile_(POINTER) 17
TagFilePos_ 21
TagFilePos_(POSITION) 17
Tagging 45
TagSet_ 21
Tech Support 53
Template Registry 6
Translation 40
Troubleshooting 52
TXD 21

- U -

Upgrading 17
Utility Templates 17

- V -

Variable Reference 44, 45

- W -

What is Query By Example? 10
Wizard 25